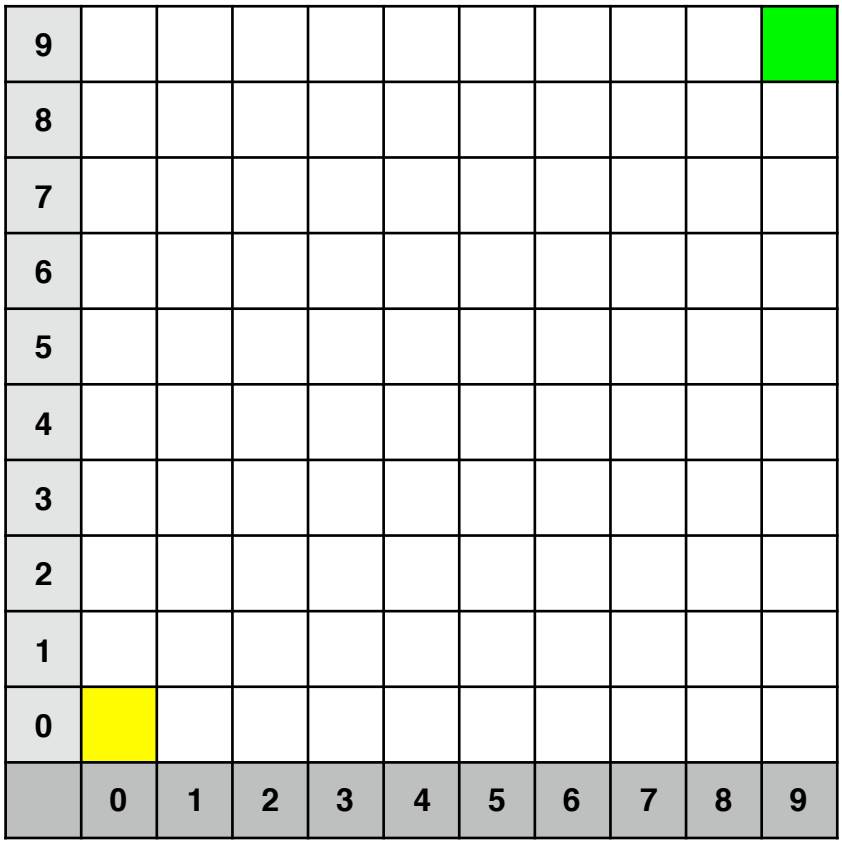
Will Skippy Get Home?



Part I: Basic Simulation

Consider the sample grid above. A lost, blind kangaroo starts in position (0,0), shaded yellow and is trying to get home, which happens to be at cell (9,9), shaded green. Skippy's strategy to find his way home is to take random hops either North, South, East or West (with equal probability), without exiting the grid of course.

The question is, will this strategy get Skippy home, or will he hop around the grid forever?

This is an example of a **Markov chain**, and we know that no matter how big the grid is, Skippy will eventually get home. The purpose of this assignment is to simulate the process and hence illustrate that Skippy gets home eventually.

This simulation can be implemented with quite a short Ruby script. However, you must use an object-oriented approach as described below.

Program Design

It is suggested to use the following classes (you may prefer a different breakdown, but it **must be object oriented**).

Die: A 4-sided die, with each side representing a direction, North, South, East or West. Also stores a hash table to keep track of the number of throws in each direction so far.

Point: A (probably mutable) point in the grid, defined by and x- and y-coordinate.

Kangaroo: Skippy is an instance of this class.

Grid: The grid that Skippy is hopping about on, can be of any dimension.

Each class should be put in a file on its own, e.g., the class Kangaroo should be in a file called **kangaroo.rb**.

Your solution should also include a script in a file called **main.rb** that:
- Instantiates the various classes;
- Starts and runs the simulation;
- Prints the final report, including the die statistics.

Assume the South-West corner of the space is (0,0) and that the grid is a square. Your solution should work for a grid of any size. Assume Skippy starts at (0,0) and that home is the North-East corner of the space at (dimension-1, dimension-1).

Input/Output Specifications

Input: Dimension of the grid. Integer, greater than or equal to 1. Check this on input.

Output: A sample program execution is given below (die stats are for example only). Your program should produce output in this format.

Sample Program Output for Part I

```
Enter dimension of the Grid (>=1):
100
Hopped to: (0, 1)
Oops, hit the boundary: (-1, 1)
Oops, hit the boundary: (-1, 1)
Hopped to: (0, 0)
Oops, hit the boundary: (-1, 0)
Hopped to: (1, 0)
...
...
Hopped to: (1, 2)
Hopped to: (0, 2)
Oops, hit the boundary: (-1, 2)
Hopped to: (1, 2)
Hopped to: (2, 2)
Finished in 18 hops!

Die statistics:
Total throws:: 53332
North: 25.2% South: 24.8% East: 25.2% West: 24.8%
```